Федеральное государственное автономное образовательное учреждение высшего образования

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ «ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Образовательная программа «Прикладная математика»
бакалавр

ОТЧЕТ
по проектной работе

Распознавание пола по аудиозаписи

Выполнили студенты гр. БПМ-195

Бикушев Глеб Дмитриевич

_____

Зубенок Павел Александрович

_____

Шевцев Марк Денисович

_____

Руководитель проекта:

_____

(*должность, ФИО руководителя проекта*)

_____ _____

(*оценка*) (*подпись*)

_____

(*дата*)

Москва 2020

Содержание

1 Вступление

Распознавание пола по аудиозаписи является важной частью систем автоматического распознавания речи и систем автоматического голосового ответа. Решение этой проблемы снижает вычислительную нагрузку на такие системы, позволяя обрабатывать человеческий голос быстрее и эффективнее.

Целью данной работы является изучение современных методов классификации данных, с выбором оптимальных параметров, чтобы найти оптимальный алгоритм определение пола человека по его голосу.

Предметом этого проекта является упрощение и ускорение обработки человеческой речи, маркировки данных. Создание систем распознавания человеческой речи Используя разные методы машинного обучения, для этого в основном используются разные классы нейронных сетей. Так как для формирования нейронных сетей вручную необходимо большое количество денег и данных, исползуются дополнительные решения.

Этот проект должен способствовать маркировке аудиоданных по полу и используемые идеи в нем, обеспечат лучшую точность при решении проблемы перевода аудиозаписей в текст.

2 Постановка задача

2.1 Неформальная постановка задачи

Используя эти аудиозаписи, создать обучающий набор данных, обучить модели на них и дать их сравнительные характеристики, предсказать пол человека с помощью тестовых аудиозаписей.

2.2 Формальная постановка задачи

Предоставлен набор аудиофайлов с расширением wav. Из этого набора случайным образом выберем 80% аудиозаписей и сформируем массив формата np.array размером $m \times n$. - обучающий образец, где первые столбцы $n-1$ являются значениями признаков для соответствующих фрагментов аудиозаписи, а n-й столбец пол человека чья это аудиозапись. Следовательно, m - число таких фрагментов. Полученный массив должен быть передан как обучающий набор к разным алгоритмам из библиотеки SciKit-Learn (далее - sklearn) и доказать гипотезу: “Независимо от того, на каком языке обучить модель, она будет верно определять пол диктора любого языка”.

3 Методы решения задачи

3.1 Описание звуковых параметров

Мел-кепстральные коэффициенты (MFCC) были выбраны в качестве параметров звука. потому что эта количественная оценка звука основана на статистической обработке большого количества данных о восприятии высоты звука человеческим слухом. Высота звука, воспринимаемого человеческим слухом, нелинейно зависит от его частоты (см.рис 1). И хотя эта зависимость не очень точна, ее довольно удобно использовать, потому что она описывается простой формулой

$$m = 1127.01048 \ln\left(1 + \frac{f}{700}\right) \tag{1}$$

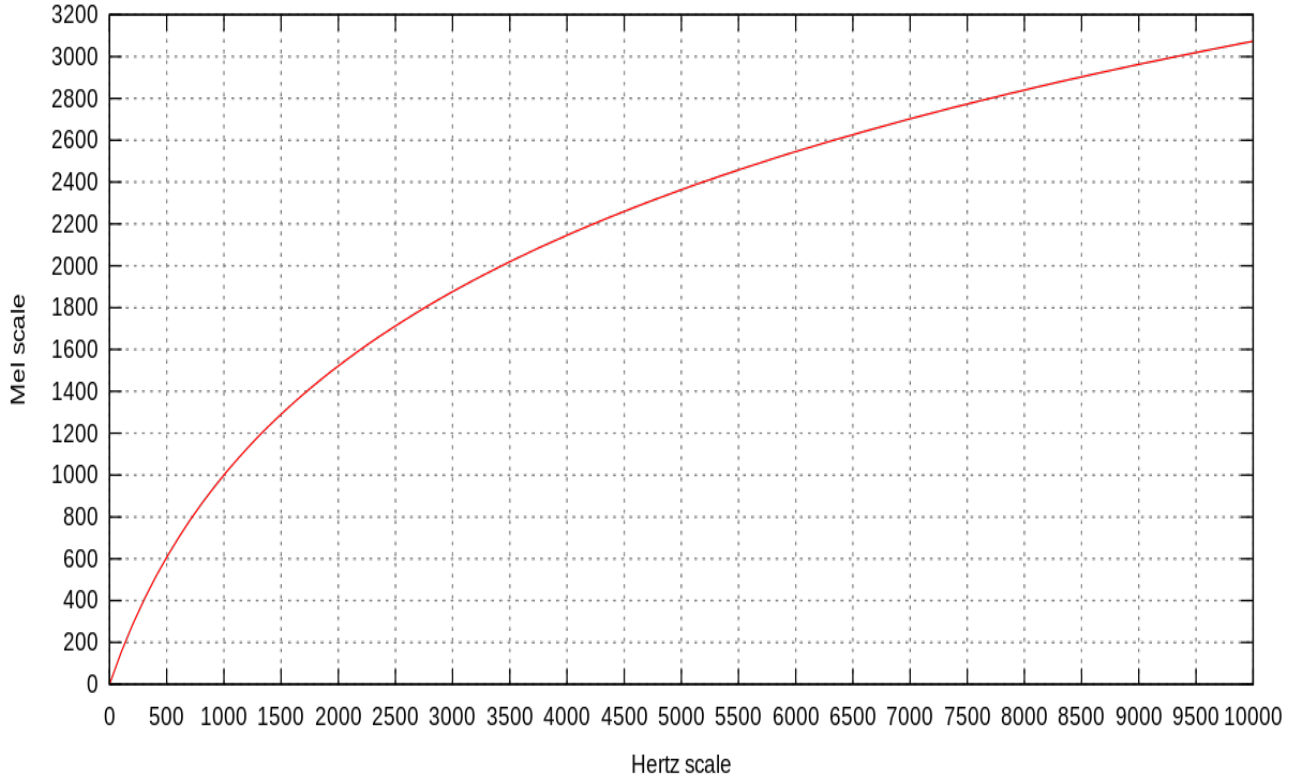

Рис. 1: График зависимости высоты звука в мелах от частоты колебаний (для чистого тона)

3.2 Рассматриваемые алгоритмы машинного обучения

Поскольку наша задача является задачей классификации, то рассматривать будем исключительно алгоритмы машинного обучения с подкреплением. Воспользуемся следующими моделями, поскольку они лучше остальных подходят под нашу задачу:

- **KNN (K Nearest Neighbors)** – метрический классификатор, основанный на оценивании сходства объектов. Классифицируемый объект относится к тому классу, которому принадлежат ближайшие к нему объекты обучающей выборки.
- **RFC (Random Forest Classifier)** – метаоценщик, который сопоставляет ряду классификаторов решающих деревьев различные подвыборки набора данных и использует усреднение для улучшение точности предсказания класса и контролирования переобучения.

3.3 Используемое программное обеспечение

1. **python_speech_features** – библиотека для работы с записями голосов, использована для извлечения MEL коэффициентов из аудиозаписей.
2. **scipy.io.wavfile** – библиотека для обработки аудиозаписей формата wav. В частности, в работе из этой библиотеки была использована функция read(), которая получает на вход wav-файл, а возвращает частоту дискретизации (в сэмплах / сек) и данные из файла wav, которые записываются в массив NumPy
3. **Anaconda** – бесплатный дистрибутив языков Python и R для научных вычислений (анализ данных, машинное обучение и приложения, обработка данных, предиктивный анализ, и так далее).

4. **Pandas** – библиотека для работы с такими объектами, как Series и DataFrame (представляют собой таблицы данных). Объект DataFrame (таблицу) можно очень удобно перевести в формат csv и работать с отдельно с этим файлом, а так же извлекать данные из файла csv в DataFrame

5. **SciKit-Learn** – библиотека, содержащая в себе реализацию основных алгоритмов машинного обучения и проверки их на точность.

6. **Pickle** – встроенный в Python модуль, использованный для сохранения наилучших моделей по результатам обучения.

7. **Matplotlib.pyplot** – библиотека для построения необходимых по заданию графиков, визуализации данных.

8. **NumPy** – библиотека, упрощающая и ускоряющая работу с вычислениями. Также необходим для полноценной работы с остальными модулями, позволяет удобно хранить массивы большой базы данных и обращаться к ним

9. **speech_recognition** и **pyaudio** – библиотеки, необходимые для записи голоса непосредственно в программе

4 Решение поставленной задачи

4.1 Создание базы данных

Для реализации машинного обучения (в нашем случае речь идёт об обучении "с учителем поскольку мы используем классификаторы) необходима база помеченных данных, в которой важная часть признаков уже разделена на отдельные категории или классы. Поэтому создадим такую базу данных, которая состоит из извлеченных MEL коэффициентов из аудиозаписей. Возьмём 16 аудиозаписей (8 - мужских, 8 - женских). Прежде всего, поясним, какие параметры функции mfcc() были взяты и для чего.

- **sig** – звуковой сигнал для расчета функций.
- **rate** – частота дискретизации.
- **winlen** - длина окна анализа в секундах. По умолчанию 0,025 с (25 миллисекунд). Сделаем этот показатель на уровне 0.2 с.
- **winstep** - шаг между последовательными окнами в секундах. По умолчанию 0,01 с (10 миллисекунд). Положим это значение, равное 0.01.
- **nfilt** - количество фильтров в наборе фильтров, по умолчанию 26. Для нашей задачи подходит 26 фильтров. Оставим это значение.
- **nfft** - размер БПФ. По умолчанию 512. Нам понадобится значение 12512.
- **lowfreq** - нижняя граница полосы фильтров mel. В Гц по умолчанию 0. Такое значение нам подходит.
- **highfreq** - максимальный край полосы фильтров из mel. Для нашей задачи подойдет значение 22050 Гц.
- **preemph** - применить фильтр preemphasis с preemph в качестве коэффициента. По умолчанию 0,97.Нам не нужен данный фильтр. Выставим значение 0.

Код программы:

```
from python_speech_features import mfcc
import scipy.io.wavfile as wav
import numpy as np
import pandas as pd
from sklearn import preprocessing

def make_data(name, dfa):
    (rate, sig) = wav.read(name)
    mfcc_feat = mfcc(sig, rate,winlen=0.2, winstep=0.01,
        nfilt=26, nfft=12512, lowfreq=0,
        highfreq=22050, preemph=0)

    res_shape = np.shape(mfcc_feat)
    df = pd.DataFrame(mfcc_feat)
    if name.find("woman" ) != -1:
        df['SEX'] = pd.Series([0] * res_shape[0], index=df.index)
    elif name.find("man" ) != -1:
        df['SEX'] = pd.Series([1] * res_shape[0], index=df.index)
    else:
        print("Wrong name of the recording." )
        exit(1)
    dfa = dfa.append(df)
    return dfa

# Создаем пустой датафрейм, который и будем передавать функциям
df_all = pd.DataFrame()

i = 1
while i < 9:
     df_all = make_data("./trening/man.wav" .format(i), df_all)
     df_all = make_data("./trening/woman.wav" .format(i), df_all)
     i += 1

# переводим полученный датафрейм в формат cv
df_all.to_csv("filedata.csv" )

# нормируем данные файла filedata.csv
min_max_scaler = preprocessing.MinMaxScaler()
np_scaled = min_max_scaler.fit_transform(df_all)
df_normalized = pd.DataFrame(np_scaled)
df_normalized.to_csv("filedata_norm.csv" )
```

На выходе получаем два файла формата csv : "filedata.csv" и "filedata_norm.csv" ("filedata.csv" , данные которого отнормированы)

4.2 Обучение нейронной сети классификатором KNN

Код программы:

```
import numpy as np
import pandas as pd
import pickle
from sklearn.model_selection import GridSearchCV
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import precision_score

frame = pd.read_csv("filedata_norm.csv" ,delimiter=',', index_col=0)

x_train = frame.iloc[:, :13].to_numpy()
y_train = frame.iloc[:, 13].to_numpy()

tuned_parametres ={
    "weights": np.array(["uniform "distance"]),
    "algorithm": np.array(["ball_tree "kd_tree "brute"]),
    "p": np.array([2, 12]),
}

model = GridSearchCV(KNeighborsClassifier(), tuned_parametres, cv=14)
# cv – количество блоков кросс-валидации
model.fit(x_train, y_train)

best_model = model.best_estimator_
y_true = y_train
y_pred = best_model.predict(x_train)
precision = precision_score(y_true, y_pred, average='weighted')
print(precision)
if precision >= 0.95:
    with open("knn_params.pickle" , "wb" ) as fout:
        pickle.dump(best_model.get_params(), fout)
        fout.close()
    with open("knn_best_model.sav" , "wb" ) as fout:
        pickle.dump(best_model, fout)
        fout.close()
```

После компиляции создаются 2 файла : "knn_params.pickle" и "knn_best_model.sav" , которые хранят информацию о наилучшей модели в результате обучения.

4.3 Обучение нейронной сети классификатором RFC

Код программы:

```
import pandas as pd
import numpy as np
import pickle
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import precision_score

frame = pd.read_csv("filedata_norm.csv" , delimiter=',', index_col=0)

x_train = frame.iloc[:, :13].to_numpy()
y_train = frame.iloc[:, 13].to_numpy()

tuned_parametres = {
    "n_estimators": np.array([50, 100, 150]),
    "criterion": np.array(["gini "entropy"]),
    "max_features": np.array(["sqrt "log2 None])
}

model = GridSearchCV(RandomForestClassifier(), tuned_parametres, cv=14)

model.fit(x_train, y_train)
best_model = model.best_estimator_
y_true = y_train
y_pred = best_model.predict(x_train)
precision = precision_score(y_true, y_pred, average='weighted')
print(precision)

if precision >= 0.95:
    with open("rfc_params.pickle"  ,"wb" ) as fout:
        pickle.dump(best_model.get_params(), fout)
        fout.close()
    with open("rfc_best_model.sav"  ,"wb" ) as fout:
        pickle.dump(best_model, fout)
        fout.close()
```

После компиляции создаются 2 файла : "rfc_params.pickle" и "frc_best_model.sav" , которые хранят информацию о наилучшей модели в результате обучения.

4.4 Основная программа, используемая для тестирования моделей

Она состоит из функции make_data, которая используется для извлечения из аудиозаписи mel - коэффициентов.

Далее идёт функция voice_to_numpy, которая нормирует mel - коэффициенты и записывает получившийся результат в массив NumPy :

```
def voice_to_numpy(name):
    frame = pd.DataFrame()
    frame = make_data(name, frame)
    min_max_scaler = preprocessing.MinMaxScaler()
    np_scaled = min_max_scaler.fit_transform(frame)
    frame_normalized = pd.DataFrame(np_scaled)
    data = frame_normalized.to_numpy()
    return data
```

Затем выбрав, один из двух классификаторов извлекаем содержимое лучшей модели формата pickle:

```
if key == 1:
    with open("knn_best_model.sav"  , "rb" ) as fin:
        model = pickle.load(fin)
elif key == 2:
    with open("rfc_best_model.sav" ,"rb" ) as fin:
        model = pickle.load(fin)
```

В завершении считаем среднее арифмтическое по столбцам предсказания пола с помощью функции библиотеки NumPy "mean()" :

```
result = np.mean(model.predict(data))
if result < 0.5:
    print("Женщина" )
    print(result)
elif result > 0.5:
    print("Мужчина" )
    print(result)
else:
    print("Неопределенно." )
    print(result)
```

4.5 Сравнение классификаторов KNN и RFC

Входные данные (аудиозапись русских дикторов монозвука с расширением wav) представляют собой временные ряды - последовательность измерений в течение каждого измерения, характеризуется соответствующим вектором коэффициентов MEL. Рассмотрим

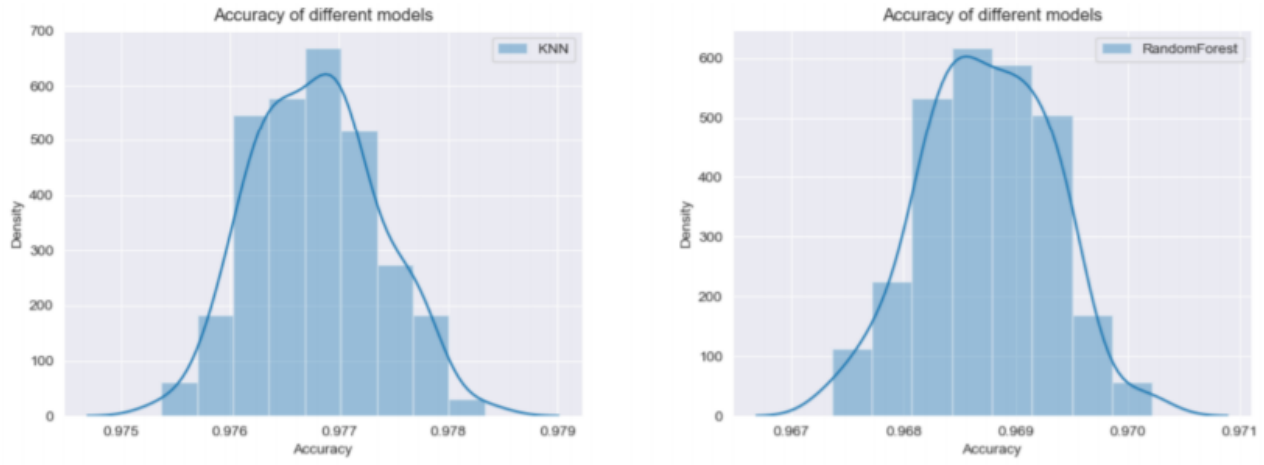

Рис. 2: Сравнительные графики зависимости плотности вероятности точности модели от её значения (за 100 запусков)

эффективность алгоритмов машинного обучения и построим сравнительные графики по стандартным параметрам с помощью библиотеки Matplotlib.

Из рис. 2 можно наблюдать, что наши модели KNN и RFC показывают точность порядка 97%, что является отличным показателем для машинного обучения.

5 Тесты

В предыдущем пункте мы сравнивали 2 алгоритма (knn и rfc). Входные данные были аудиозаписи на русском языке.

Ниже представлена таблица где входные данные были аудиозаписи на разных языках.

Как можно заметить, хоть и с небольшой погрешностью, но разработанная модель верно определяет пол диктора иностранного языка, не смотря на то, что была обучена на записях русской речи.

номер записи	язык диктора	классификатор	пол	результат
1	английский	KNN	муж	жен (0.243)
1	английский	RFC	муж	жен (0.342)
2	английский	KNN	жен	жен (0.267)
2	английский	RFC	жен	жен (0.187)
3	немецкий	KNN	муж	муж (0.509)
3	немецкий	RFC	муж	муж (0.564)
4	немецкий	KNN	жен	жен (0.438)
4	немецкий	RFC	жен	жен (0.213)
5	французский	KNN	муж	муж (0.611)
5	французский	RFC	муж	муж (0.506)
6	французский	KNN	жен	жен (0.335)
6	французский	RFC	жен	жен (0.212)
7	испанский	KNN	муж	муж (0.619)
7	испанский	RFC	муж	муж (0.758)
8	испанский	KNN	жен	жен (0.473)
8	испанский	RFC	жен	жен (0.292)
9	вьетнамский	KNN	муж	муж (0.561)
9	вьетнамский	RFC	муж	муж (0.230)
10	вьетнамский	KNN	жен	жен (0.387)
10	вьетнамский	RFC	жен	жен (0.244)
11	грузинский	KNN	муж	муж (0.681)
11	грузинский	RFC	муж	муж (0.565)
12	грузинский	KNN	жен	жен (0.438)
12	грузинский	RFC	жен	муж (0.723)
13	айзербаджанский	KNN	муж	муж (0.554)
13	айзербаджанский	RFC	муж	муж (0.703)
14	айзербаджанский	KNN	жен	жен (0.476)
14	айзербаджанский	RFC	жен	жен (0.408)

6 Результат

Была разработана программа, которая определяет пол диктора по аудиозаписи. Была проверена и подтверждена гипотеза: "Независимо от того, на каком языке обучить модель, она будет верно определять пол диктора любого языка" . Найти код проекта, а также самому проверить исправность работы нашей модели, записав свой голос сразу в программе можно по этой ссылке, перейдя к GitHub репозиторий.

Список литературы

[1] Мел-кепстральные коэффициенты (MFCC) и распознавание речи // article from Habr.com // URL: https://habr.com/ru/post/140828/

[2] Классификатор kNN // article from Habr.com.// URL: https://habr.com/ru/post/149693/

[3] Реализация и разбор алгоритма «случайный лес» на Python // article from tproger.ru// URL: https://tproger.ru/translations/python-random-forest-implementation/

[4] Welcome to python_speech_features's documentation! // URL: https://python-speech-features.readthedocs.io/en/latest/